Discovering Mis-Categorized Entities

Shuang Hao[†] Nan Tang[‡] Guoliang Li[†] Jianhua Feng[†]
[†]*Department of Computer Science, Tsinghua University, China* [‡]*Qatar Computing Research Institute, HBKU, Qatar*
{*haos13@mails., liguoliang@, fengjh@*}*tsinghua.edu.cn, ntang@hbku.edu.qa*

***Abstract*—Entity categorization – the process of grouping entities into categories – is an important problem with a great many applications. Unfortunately, in practice, many entities are mis-categorized, such as Google Scholar and Amazon products. In this paper, we study the problem of *discovering mis-categorized entities* from a given group of entities. This problem is inherently hard: all entities within the same group have been "well" categorized by *state-of-the-art* solutions. Apparently, it is nontrivial to differentiate them. We propose a novel *rule-based framework* to solve this problem. It first uses positive rules to compute disjoint partitions of entities, where the partition with the largest size is taken as the correctly categorized partition, namely the *pivot partition*. It then uses negative rules to identify mis-categorized entities in other partitions that are *dissimilar* to the entities in the pivot partition. We describe optimizations on applying these rules, and discuss how to generate positive/negative rules. Extensive experimental results on real-world datasets show the effectiveness of our solution.**

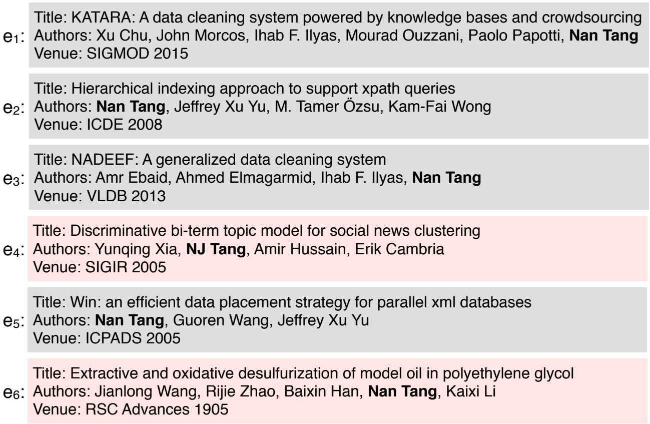
I. Introduction

Categorizing entities into sensible groups is a fundamental mode for many applications. Nevertheless, mis-categorized entities are pervasive, *e.g.,* there exist others' publications in many researchers' Google Scholar pages. A practical problem is: *how to (automatically) discover them?*

Example 1: [Mis-Categorized Google Scholar Entities.] Consider *Nan Tang*'s sample Google Scholar entities in Figure 1. Each entity has three attributes: `Title`, `Authors` and `Venue`. Two entities, e_4 and e_6, are mis-categorized – they do not belong to the *Nan Tang* at QCRI. The e_4[`Authors`] does not contain a valid name, and the *Nan Tang* in e_6[`Authors`] is a different person working in chemistry.
[Mis-Categorized Amazon Product Entities.] Consider sample entities in Amazon's product category "*Router*", shown in Figure 2. Each entity has four attributes: `Asin` is the ID of product, `Title` is its name, `Also_viewed` contains a list of products viewed together with it, and `Description` describes its features. The entity o_3 is mis-categorized, since it should be in category "*Adapter*" instead of "*Router*". □

Entity categorization (EC) is related to, but different from, entity matching (EM). EM is to determine if two entities are the same, which is often solved by comparing aligned attributes symbolically (or syntactically). EC groups entities more conceptually (or semantically), *e.g.,* e_1 and e_2 in Figure 1 are not symbolically similar in either `Title`, `Authors`, or `Venue`.

Discovering mis-categorized entities is hard – all entities that are categorized into the same group already use all attributes of the entities, how can we use the same set of attributes (or evidence) to discover mis-categorized entities?



e_1: Title: KATARA: A data cleaning system powered by knowledge bases and crowdsourcing
Authors: Xu Chu, John Morcos, Ihab F. Ilyas, Mourad Ouzzani, Paolo Papotti, **Nan Tang**
Venue: SIGMOD 2015

e_2: Title: Hierarchical indexing approach to support xpath queries
Authors: **Nan Tang**, Jeffrey Xu Yu, M. Tamer Özsu, Kam-Fai Wong
Venue: ICDE 2008

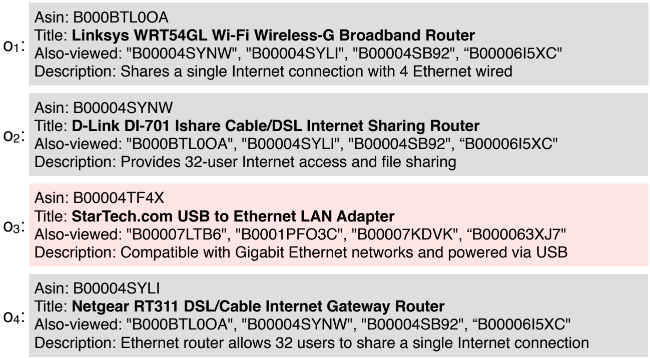
e_3: Title: NADEEF: A generalized data cleaning system
Authors: Amr Ebaid, Ahmed Elmagarmid, Ihab F. Ilyas, **Nan Tang**
Venue: VLDB 2013

e_4: Title: Discriminative bi-term topic model for social news clustering
Authors: Yunqing Xia, **NJ Tang**, Amir Hussain, Erik Cambria
Venue: SIGIR 2005

e_5: Title: Win: an efficient data placement strategy for parallel xml databases
Authors: **Nan Tang**, Guoren Wang, Jeffrey Xu Yu
Venue: ICPADS 2005

e_6: Title: Extractive and oxidative desulfurization of model oil in polyethylene glycol
Authors: Jianlong Wang, Rijie Zhao, Baixin Han, **Nan Tang**, Kaixi Li
Venue: RSC Advances 1905

Figure 1. Sample Google Scholar Entities of Nan Tang



o_1: Asin: B000BTL0OA
Title: **Linksys WRT54GL Wi-Fi Wireless-G Broadband Router**
Also-viewed: "B00004SYNW", "B00004SYLI", "B00004SB92", "B00006I5XC"
Description: Shares a single Internet connection with 4 Ethernet wired

o_2: Asin: B00004SYNW
Title: **D-Link DI-701 Ishare Cable/DSL Internet Sharing Router**
Also-viewed: "B000BTL0OA", "B00004SYLI", "B00004SB92", "B00006I5XC"
Description: Provides 32-user Internet access and file sharing

o_3: Asin: B00004TF4X
Title: **StarTech.com USB to Ethernet LAN Adapter**
Also-viewed: "B00007LTB6", "B0001PFO3C", "B00007KDVK", "B000063XJ7"
Description: Compatible with Gigabit Ethernet networks and powered via USB

o_4: Asin: B00004SYLI
Title: **Netgear RT311 DSL/Cable Internet Gateway Router**
Also-viewed: "B000BTL0OA", "B00004SYNW", "B00004SB92", "B00006I5XC"
Description: Ethernet router allows 32 users to share a single Internet connection

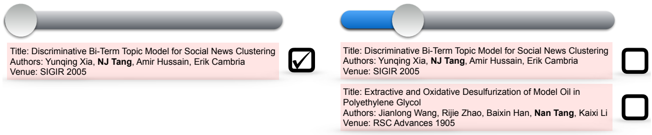
Figure 2. Sample Entities in Amazon's Router Category

We propose a rule-based framework using positive and negative rules to discover mis-categorized entities. Positive rules are used conservatively to find disjoint partitions, such that the entities within the same partition should be categorized together. The partition with the largest size is called the *pivot partition*, under the practical assumption that the largest partition is correct. Negative rules are used to compare other partitions with the pivot partition to discover dissimilar entities as mis-categorized entities.

Example 2: Consider the entities in Figure 1, and the following positive rules (φ^+) and negative rules (ϕ^-). We use the terms "similar"/"dissimilar" to indicate that two entities should/shouldn't be in the same category.

φ_1^+ : *Two entities are similar, if they have* $\geqslant 2$ *common* `Authors`
φ_2^+ : *Two entities are similar, if they have overlapping in* `Authors` *and their* `Venue` *are in the same field.*

[1] Guoliang Li is the corresponding author.


Title: Discriminative Bi-Term Topic Model for Social News Clustering
Authors: Yunqing Xia, **NJ Tang**, Amir Hussain, Erik Cambria
Venue: SIGIR 2005 ☑

(a) Author

Title: Discriminative Bi-Term Topic Model for Social News Clustering
Authors: Yunqing Xia, **NJ Tang**, Amir Hussain, Erik Cambria
Venue: SIGIR 2005 □

Title: Extractive and Oxidative Desulfurization of Model Oil in Polyethylene Glycol
Authors: Jianlong Wang, Rijie Zhao, Baixin Han, **Nan Tang**, Kaixi Li
Venue: RSC Advances 1905 □

(b) Author and Venue

Figure 3. Tuning Discovered Mis-categorized Entities Using a Scrollbar

ϕ_1^- : *Two entities are dissimilar, if they have no common* `Author`.
ϕ_2^- : *Two entities are dissimilar, if they have* $\leqslant 1$ *common author in* `Authors` *and their* `Venue` *are in different fields.*

(1) *Positive rules* are used in a disjunctive fashion as $\varphi_1^+ \vee \varphi_2^+$: two entities belong to the same category if either φ_1^+ or φ_2^+ is *true*. Also, we assume the *transitivity* of categorization: entities (e_i, e_j) and (e_j, e_k) are categorized together implies that (e_i, e_k) are also categorized together. We can find three partitions $P_1 : \{e_1, e_2, e_3, e_5\}$, $P_2 : \{e_4\}$, $P_3 : \{e_6\}$. For instance, e_1 and e_2 are categorized together based on φ_2^+: they have an overlapping author *Nan Tang*, and their venues *SIGMOD* in e_1 and *ICDE* in e_2 are in the same field (see ontology-based similarity in Section II).

(2) The partition P_1 is considered as the pivot partition, because it has the largest size.

(3) *Negative rules* are used either individually *e.g.,* ϕ_1^-, or in a disjunction *e.g.,* $\phi_1^- \vee \phi_2^-$: two entities belong to different categories if either ϕ_1^- or ϕ_2^- is *true*. ϕ_1^- discovers e_4 as one mis-categorized entity because e_4 does not have overlapping in `Authors` with any entity in P_1. $\phi_1^- \vee \phi_2^-$ further discovers e_6, since e_6 only has one common author with the entities in P_1 and its `Venue` is in the field of *Chemical Sciences*, which is different from the field *Computer Science* in P_1. Negative rules are used in sequence – we first use ϕ_1^-, then $\phi_1^- \vee \phi_2^-$, and so on, if more negative rules are available. We provide a scrollbar for the user to confirm the mis-categorized entities from either ϕ_1^- or $\phi_1^- \vee \phi_2^-$, shown in Figure 3. □

We can see from Example 2 that: (i) Positive rules need to be conservative, so as not to include mis-categorized entities in the pivot partition; (ii) We apply multiple negative rules based on the "*one does not fit all*" philosophy – no single negative rule can perfectly discover mis-categorized entities for every given group; and (iii) The purpose to apply a sequence of negative rules, which ensures that their outputs are in a monotonic fashion, is easy for users to operate.

Contributions. We summarize our contributions below.

(1) We define the new problem of *discovering mis-categorized entities* (Section II).

(2) We present the solution overview of a novel *rule-based framework* to tackle the studied problem (Section III).

(3) We devise signature-based algorithms to improve the performance of applying positive/negative rules (Section IV).

(4) We describe effective algorithms to generate positive and negative rules from examples (Section V).

(5) We ran extensive experiments on real-world datasets to show that our framework can efficiently discover mis-categorized entities with high accuracy (Section VI).

Moreover, we have released a Google Chrome extension called GSCleaner for cleaning Google Scholar pages (https://github.com/TsinghuaDatabaseGroup/googlescholar), which will also be demonstrated in ICDE 2018 [1].

Related Work. Entity Matching (EM) finds pairs of entities that refer to the same real-world object. EM rules have been widely used [2], and there are solutions to learn EM rules by examples [3], [4] Moreover, there are also machine learning based EM methods [5], [6], and crowd-based solutions [7]. Collective entity resolution [8] uses additional information to match entities by an agglomerative clustering algorithm.

As mentioned earlier, entity categorization (EC) differs from EM in that entities within the same category typically refer to different real-world objects. Consequently, the traditional wisdom of inferring whether entities are the same cannot be directly applied to our studied problem.

Named Entity Disambiguation (NED) is the task of disambiguating named entities mentioned in text of data and link them to their corresponding entries [9]–[11]. NED requires more efforts from NLP to handle unstructured data.

NED aims to distinguish entities that have the same name but refer to different real-world entities. In addition to disambiguation, we also identify entities that have different names (or references) but should be in the same category.

Clustering Algorithms. Another related topic is clustering, where thousands of clustering algorithms have been published in the last 50+ years [12]. Perhaps the most widely used is the k-means clustering [13]. Clearly, finding a "perfect" clustering algorithm that computes two partitions for a group, one for correctly categorized entities and the other for mis-categorized entities, is likely to fail, as will be empirically verified in Section VI.

II. Problem and Notation

Problem. Given a group $\mathbf{G}$ of entities that have been categorized together by existing algorithms, the problem of *discovering mis-categorized entities* is to find the proper subset $\mathbf{G}' \subset \mathbf{G}$ that should not belong to this group.

Next, we define the notations that are used in this paper.

Entities. An *entity* e is defined over a multi-valued relation $R(A_1, \ldots, A_m)$ of m attributes (*a.k.a.* fields or features). Each attribute A_i of an entity can take a list of values. We write $e[A_i]$ the attribute value of e on attribute A_i.

Groups. A *group* $\mathbf{G}$ is a set of entities $\{e_1, \ldots, e_n\}$.

Example 3: [Entities.] Consider entities in Figure 1, which are defined over schema $(\texttt{Title}, \texttt{Authors}, \texttt{Venue})$. Entity e_1 has three attributes, and $e_1[\texttt{Authors}]$ has multi values.

[Groups.] Nan's group is shown in Figure 1. □

Positive Rules. A *positive rule* $\varphi^+(e, e')$ is a *conjunction* of predicates: $\varphi^+(e, e') = \bigwedge_{A_i \in R} f_i(A_i) \geqslant \theta_i$, where $f_i(A_i)$

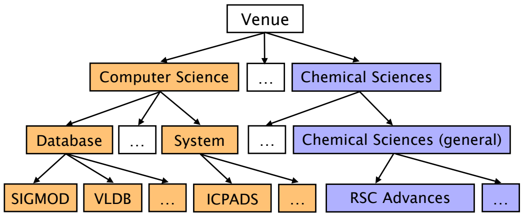


Figure 4. Google Scholar Metrics

is a similarity function and θ_i is a threshold. $\varphi^+(e, e')$ is evaluated to be *true* if all predicates $f_i(A_i) \geqslant \theta_i$ return *true*, **indicating the two entities e and e' are "similar" and should be categorized together**. Otherwise, $\varphi^+(e, e')$ returns *false*, if any of the predicates returns *false*, indicating we *do not know* whether they should be in the same category.

Negative Rules. A *negative rule* $\phi^-(e, e')$ is defined similarly: $\phi^-(e, e') = \bigwedge_{A_i \in R} f_i(A_i) \leqslant \sigma_i$. $\phi^-(e, e')$ is evaluated to be *true* if all predicates $f_i(A_i) \leqslant \sigma_i$ return *true*, **indicating e and e' are "dissimilar" and should not be categorized together**. Otherwise, $\phi^-(e, e')$ returns *false*, if any of the predicates returns *false*, indicating we *do not know* whether they should be in different categories.

We will simply write φ^+ (resp. ϕ^-) as a positive (resp. a negative) rule, when (e, e') is clear from the context.

Similarity Functions. We consider three types of similarity functions f to quantify the similarity between two values.
(i) Set-based. It first splits each value into a set of tokens and then utilizes the set-based similarity to quantify the similarity, such as overlap and Jaccard similarity.
(ii) Character-based. It measures the similarity between two values based on character transformations, like edit distance.

The above similarity functions have a common limitation that they mainly use the symbolic (or textual) information, while ignoring the semantics, which is important to the entity categorization problem. To this purpose, we propose to use ontology to capture the semantic similarity. Many enterprises are building their own ontologies, such as Google and Amazon. We can use them directly, or build an ontology using existing knowledge-base construction techniques [14].

(iii) Ontology-based. Ontology is usually modeled by a tree structure, *e.g.,* the ontology for venues of publications provided by Google Scholar Metric[2] is shown in Figure 4.

Given two entities, we first map them to tree nodes[3]. The ontology similarity is computed based on their lowest common ancestor (LCA), which is formally defined as follows.

Ontology Similarity. Given two entities e and e', let their mapping nodes on the ontology tree be n and n', respectively. Their *ontology similarity* is defined as: $\frac{2|\texttt{LCA}(n,n')|}{|n|+|n'|}$, where $\texttt{LCA}(n, n')$ is the lowest common ancestor of n and n', and $|n|$ is the depth of node n in the tree (the depth of the root is 1). Two entities are *similar* if their similarity is larger than a threshold τ.

```
Algorithm 1: DIME: Rule-based Framework
Input: a group $\mathbf{G} = \{e_1, ..., e_n\}$, a set of positive rules
       $\{\varphi_1^+, \dots \varphi_x^+\}$, a set of negative rules $\{\phi_1^-, \dots \phi_y^-\}$
Output: mis-categorized entities $\mathbf{G}^-$
// Step 1: Computing Disjoint Partition Set $\mathbf{P}$
1  Construct a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \mathbf{G}, \mathcal{E} = \varnothing$;
2  for each entity pair $(e, e') \in \mathbf{G} \times \mathbf{G}$ do
3      if $\exists \varphi_i^+$ such that $\varphi_i^+(e, e')$ returns true then
4          $\mathcal{E} \leftarrow \mathcal{E} \cup \{(e, e')\}$;
5  Compute the connected components of $\mathcal{G}$;
6  Let $\mathbf{P}$ denote the set of connected components in $\mathcal{G}$;
// Step 2: Identifying The Pivot Partition $P^*$
7  Let $P^* \in \mathbf{P}$ (the one with the largest size) be pivot partition;
// Step 3: Discovering Mis-Categorized Entities $\mathbf{G}^-$
8  for each partition $P \in \mathbf{P} \backslash \{P^*\}$ do
9      if $\exists (e \in P, e^* \in P^*, \phi^- \in \Sigma^-)$, $\phi^-(e, e^*)$ returns true then
10         $\mathbf{G}^- \leftarrow \mathbf{G}^- \cup P$;
11 return $\mathbf{G}^-$
```

Example 4: [Ontology Similarity.] Consider the tree structure of venues in Figure 4, and two nodes *SIGMOD* and *VLDB*. They have rather low string similarity. However, they have a high ontology similarity $\frac{3}{4}$, because their depths are both 4 and their LCA is *Database* with depth 3. □

The rules in Example 2 can be formulated as follows.

$\varphi_1^+ : f_{ov}(\texttt{Authors}) \geqslant 2$
$\varphi_2^+ : f_{ov}(\texttt{Authors}) \geqslant 1 \wedge f_{on}(\texttt{Venue}) \geqslant 0.75$
$\phi_1^- : f_{ov}(\texttt{Authors}) = 0$
$\phi_2^- : f_{ov}(\texttt{Authors}) \leqslant 1 \wedge f_{on}(\texttt{Venue}) \leqslant 0.25$

where f_{ov} denotes overlap similarity (common authors in the above rules) and f_{on} indicates ontology similarity.

III. A Rule-based Framework

Our framework DIME is described in Algorithm 1 and depicted in Figure 5. In a nutshell, it first uses a set of positive rules $\Sigma^+ = \{\varphi_1^+, \cdots, \varphi_x^+\}$ as a disjunction (*i.e.,* $\varphi_1^+ \vee \cdots \vee \varphi_x^+$) on a group of entities $\mathbf{G} = \{e_1, \dots, e_n\}$ to compute disjoint partitions (step 1; lines 2-6). The partition with the largest size is the *pivot partition* (line 7). Given a set of negative rules $\Sigma^- = \{\phi_1^-, \cdots, \phi_y^-\}$, we either apply the first rule ϕ_1^-, or jointly use ϕ_1^- with the other negative rules in sequence as $\phi_1^- \vee \phi_2^-$, $\phi_1^- \vee \phi_2^- \vee \phi_3^-$, and so on, to discover mis-categorized entities (lines 8-11).

Step 1: Computing Disjoint Partitions. Positive rules are used to group entities into partitions. Two cases are considered to put entities e and e' in the same partition. (i) e and e' satisfy a positive rule; or (ii) e and e' satisfy transitivity – there exists another entity e'' such that both (e, e'') and (e', e'') match.

For case (i), we enumerate every entity pair (e, e') and every rule φ^+, and check whether $\varphi^+(e, e')$ returns true. For case (ii), we first construct a graph, where the vertices are entities and edges are entity pairs that satisfy a positive rule

[2] https://scholar.google.com/citations?view_op=top_venues.
[3] Here we use exact string matching for example. We can also use approximate matching based on similarity functions.

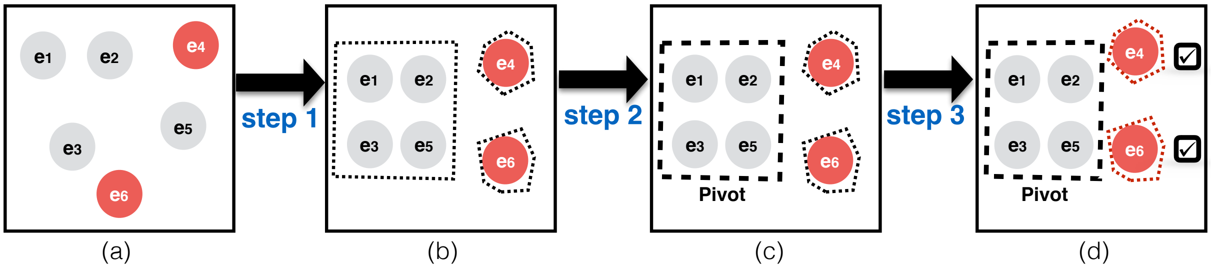


Figure 5. Solution Overview: A Rule-based Framework

(computed from case (i)), and then compute its connected components. Clearly, the entities in the same connected component satisfy the transitivity and form a partition.

The complexity of checking whether an entity pair satisfies a positive rule depends on the similarity functions used in the rule. For overlap, the complexity of computing the similarity is $\mathcal{O}(|e| + |e'|)$, where $|e|$ is the size of e. For edit distance, the complexity of computing the similarity is $\mathcal{O}(\theta \min(|e|, |e'|)$, where θ is the similarity threshold and $|e|$ is the length of e. For ontology, the complexity of computing the similarity is $\mathcal{O}(|d_e| + |d_{e'}|)$, where $|d_e|$ is the depth of e's corresponding node in the tree structure. For ease of presentation, suppose the complexity of checking whether an entity pair satisfies a positive rule is $\mathcal{O}(v)$. The time complexity of checking every entity pair and every positive rule is $\mathcal{O}(n^2 v|\Sigma^+|)$, where n is the number of entities and $|\Sigma^+|$ is the number of positive rules. The time complexity of computing connected components (*e.g.,* by a depth-first traversal) is the number of vertices and edges in the graph. Thus the overall complexity is $\mathcal{O}(n^2 v|\Sigma^+|)$.

Step 2: Identifying the Pivot Partition. The pivot partition $P^* \in \mathbf{P}$ is the one with the largest size, which is treated as the correctly categorized partition.

Step 3: Discovering Mis-Categorized Entities. Given the set $\mathbf{P}$ of partitions and the pivot partition P^*, we use the negative rules to mark whether another partition $P \in \mathbf{P}\backslash\{P^*\}$ is a wrongly categorized partition, such that if P is, all entities in P are reported as mis-categorized. To discover the mis-categorized partitions, we enumerate every entity pair $(e^* \in P^*, e \in P)$, and every negative rule $\phi^- \in \Sigma^-$, if $\phi^-(e, e^*)$ returns true, we mark P as a wrongly categorized partition. The complexity of this algorithm is $\mathcal{O}(n^2 v|\Sigma^-|)$.

The GUI for Scrolling with Multiple Negative Rules. Depending on user's capacity, we provide user with a GUI to check manually the outputs of multiple combinations of negative rules, which works pretty well for Google Scholar [1]. The benefits are as follows. (1) The positive/negative rules are provided – the user does not need to know how they are generated (see Section V for more details). (2) It is cheaper to manually check our suggested mis-categorized entities than checking the entire group, since the number of mis-categorized entities is typically much smaller than the total number of entities in a group; that is, $|\mathbf{G}^-| \ll |\mathbf{G}|$.

Example 5: [Rule-based Framework.] Consider entities in Example 1, which are shown in Figure 5(a).

(1) Consider the positive rules φ_1^+ and φ_2^+ in Example 2. Using $\varphi_1^+ \vee \varphi_2^+$ we can find three partitions $P_1 : \{e_1, e_2, e_3, e_5\}$, $P_2 : \{e_4\}$ and $P_3 : \{e_6\}$, shown in Figure 5(b).

(2) The pivot partition $P^* = P_1$ is surrounded by the dashed rectangle in Figure 5(c).

(3) Using negative rules $\phi_1^- \vee \phi_2^-$ in Example 2, we can discover mis-categorized entities e_4 and e_6, as shown in Figure 5(d) (see Example 2 for more explanations about how negative rules are used and see Figure 3 for the usage of a scrollbar that applies either ϕ_1^- or $\phi_1^- \vee \phi_2^-$). □

Also, it is practical to make the following assumptions.

ASSUMPTIONS. (1) The *pivot partition* – the partition with the largest size – only contains truly categorized entities. The rationality is that compared with other partitions with much smaller sizes, the pivot partition is more likely to be correct; that is, existing algorithms that produce these groups are not that bad. (2) We assume the *transitivity*: If e matches e' and e' matches e'', then e and e'' match. □

IV. SIGNATURE-BASED FAST SOLUTION

The naïve method of applying positive/negative rules is by enumerating all pairs of entities – clearly an expensive solution. We propose a fast signature-based framework $\texttt{DIME}^+$ (Section IV-A). We also study how to generate signatures (Section IV-B). Finally, we present algorithms for processing positive rules (Section IV-C) and negative rules (Section IV-D).

A. A Signature-Based Framework

We propose a *filter-verification* framework to efficiently find entity pairs satisfying given positive or negative rules.

Signatures of Entities. Signatures are substrings (or ancestor nodes in the ontology hierarchy) of entities that can be used to prune pairs of entities that cannot satisfy a positive/negative rule. Intuitively, two entities match, only if they share enough common signatures.

Efficient Algorithms for Positive Rules. The "*filter*" step: we generate signatures for each entity *w.r.t.* each positive rule: If two entities satisfy a positive rule, they must share a common signature. Thus we can take the pairs of entities that share common signatures as candidate pairs (and other pairs that do not share common signatures can be safely pruned). To find such candidate pairs, we build a signature-based inverted index for each positive rule, which keeps a mapping from a signature to an inverted list of entities that contain the signature. Then the entity pair on each inverted list is taken as a candidate pair. We will describe how to generate signatures in Section IV-B.

The "*verification*" step: we verify the candidate pairs by computing their real similarities. We use the transitivity to avoid computing the similarities of unnecessary pairs: If (e, e') and (e', e'') are categorized together, we can infer that (e, e'') should also be categorized together without further verifying (e, e''), which will be elaborated in Section IV-C.

Efficient Algorithms for Negative Rules. A negative rule is defined on top of partitions. A partition P is dissimilar with the pivot partition P^* if there exists a pair of entities $(e \in P, e^* \in P^*)$ and a negative rule ϕ^- such that $\phi^-(e, e^*)$ returns true. To effectively check whether two partitions are dissimilar, we also utilize the signatures in the "*filter*" step: If two entities share common signatures, they may be similar; if two entities do not share any signature, they must be dissimilar. So, we generate the signature of a partition which is the union of signatures of entities in the partition. If two partitions have no common signatures, they satisfy the negative rule; otherwise, we verify the pairs of entities across these two partitions.

The "*verification*" step: we first verify the entity pair with large probability to be dissimilar, because once we find a pair satisfying the negative rule, we do not need to verify the other pairs, which will be discussed in Section IV-D.

A Signature-based Algorithm. We present DIME$^+$ in Algorithm 2. It first utilizes the signatures to compute disjoint partitions (lines 1-11 for step 1). It enumerates every entity e in $\mathbf{G}$, computes its signature g, and builds inverted list $L_{\varphi_i^+}(g)$ for positive rule φ_i^+, which keeps the list of entities that contain g (lines 1-2). It then computes the entity pairs in each inverted list as candidate pairs (lines 3-4). Next it sorts the candidate pairs and builds a graph (line 5-6). If e and e' belong to different connected components and satisfy a positive rule, we add the edge into the graph (lines 7-10). Then it computes the connected components (line 11).

It then picks the pivot partition (line 12 for step 2).

Finally, it uses the signatures to discover mis-categorized entities that are not in the pivot partition (lines 13-24 for step 3). It generates the signature set of pivot partition P^* by computing the union of signatures of entities in P^* for each negative rule ϕ_i^-, denoted by $L_{\phi_i^-}(P^*)$ (lines 13-14). For each partition P, it computes its signature set $L_{\phi_i^-}(P)$ (lines 15-17). If the signature sets of P and P^* have no overlap, P is a mis-categorized partition (lines 18-19); otherwise, (e, e^*) that $e \in P, e^* \in P^*$ is a candidate (line 20). Then it sorts candidates and checks whether each candidate satisfies a negative rule (lines 22-24). The mis-categorized entities are returned (line 25).

Algorithm 2: DIME$^+$: Signature-based Algorithm

```
   Input: a group $\mathbf{G} = \{e_1, ..., e_n\}$, a set of positive rules
          $\{\varphi_1^+, \dots \varphi_x^+\}$, a set of negative rules $\{\phi_1^-, \dots \phi_y^-\}$
   Output: mis-categorized entities $\mathbf{G}^-$
   // Step 1: Computing Disjoint Partitions $\mathbf{P}$
 1 for each signature $g$ of $e \in \mathbf{G}$ wrt $\varphi_i^+$ do
 2     $L_{\varphi_i^+}(g) \leftarrow L_{\varphi_i^+}(g) \cup \{e\}$;
 3 for two entities $e, e' \in L_{\varphi_i^+}(g)$ do
 4     $\mathcal{C} \leftarrow \mathcal{C} \cup \{(e, e')\}$;
 5 sort the candidate set $\mathcal{C}$;
 6 build a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \mathbf{G}, \mathcal{E} = \varnothing$;
 7 for each candidate $(e, e') \in \mathcal{C}$ do
 8     if $e,e'$ belong to different connect components then
 9         if $\varphi^+(e, e')$ returns true then
10             add an edge $(e, e')$ to $\mathcal{G}$;
11 $\mathbf{P} \leftarrow$ the connect components of $\mathcal{G}$;
   // Step 2: Identifying The Pivot Partition $P^*$
12 Let $P^* \in \mathbf{P}$ be the pivot partition; $\mathcal{C} \leftarrow \varnothing$;
   // Step 3: Discovering mis-categorized Entities $\mathbf{G}^-$
13 for each signature $g$ of $e \in P^*$ w.r.t. $\phi_i^-$ do
14     $L_{\phi_i^-}(P^*) \leftarrow L_{\phi_i^-}(P^*) \cup \{g\}$;
15 for each partition $P \in \mathbf{P} \backslash \{P^*\}$ do
16     for each signature $g$ of $e \in P$ wrt $\phi_i^-$ do
17         $L_{\phi_i^-}(P) \leftarrow L_{\phi_i^-}(P) \cup \{g\}$;
18     if $L_{\phi_i^-}(P) \cap L_{\phi_i^-}(P^*) = \varnothing$ then
19         $\mathbf{G}^- \leftarrow \mathbf{G}^- \cup P$;
20     else add each pair $(e, e^*)$ that $e \in P, e^* \in P^*$ into $\mathcal{C}$ ;
21 sort the candidate set $\mathcal{C}$;
22 for each candidate $(e, e^*) \in \mathcal{C}$ do
23     if $e \notin \mathbf{G}^-$ and $\phi_i^-(e, e^*)$ returns true then
24         $\mathbf{G}^- \leftarrow \mathbf{G}^- \cup P$;
25 return $\mathbf{G}^-$
```

B. Signature Generation

Signatures for Positive Rules. Given a positive rule $\varphi^+ = \bigwedge_{A_i \in R} f_i(A_i) \geqslant \theta_i$, for each predicate $f_i(A_i) \geqslant \theta_i$, we generate a signature set $\mathtt{Sig}^i_{\varphi^+}(e)$ for each entity e such that if two entities, e and e', can satisfy this predicate, they must share some common signatures, *i.e.,* $\mathtt{Sig}^i_{\varphi^+}(e) \cap \mathtt{Sig}^i_{\varphi^+}(e') \neq \varnothing$. Conversely, if two entities do not share any common signature, they must be dissimilar. Note that a positive rule φ^+ is a conjunction of predicates $\bigwedge_{A_i \in R} f_i(A_i)$, we pick one signature *w.r.t.* each predicate as the signature *w.r.t.* φ^+ for an entity e, so $\mathtt{Sig}_{\varphi^+}(e) = \{(g^1, g^2, ..., g^{|\varphi^+|}) \mid g^1 \in \mathtt{Sig}^1_{\varphi^+}(e), g^2 \in \mathtt{Sig}^2_{\varphi^+}(e), \cdots, g^{|\varphi^+|} \in \mathtt{Sig}^{|\varphi^+|}_{\varphi^+}(e)\}$, where $\mathtt{Sig}^i_{\varphi^+}(e)$ is the signature *w.r.t.* the i-th predicate.

Next we discuss how to generate the signatures $\mathtt{Sig}^i_{\varphi^+}(e)$. For set-based and character-based functions, we use existing techniques to generate signatures [15], and here we just discuss the signature generation for ontology similarity.

For ease of presentation, we use entity e and its mapping tree node n interchangeably. Given a node n, if another node n' is similar to n, we have $\frac{2|\texttt{LCA}(n,n')|}{|n|+|\texttt{LCA}(n,n')|} \geqslant \frac{2|\texttt{LCA}(n,n')|}{|n|+|n'|} \geqslant \theta$ and $|\texttt{LCA}(n,n')| \geqslant \frac{\theta|n|}{2-\theta}$. Let $\tau_n = \lceil\frac{\theta|n|}{2-\theta}\rceil$ and denote $\mathcal{A}_{\tau_n}$ by the ancestor of n at depth τ_n. We can take $\mathcal{A}_{\tau_n}$ as a signature of n. Similarly we can take $\mathcal{A}_{\tau_{n'}}$ as a signature of n'. If n and n' are similar, both $\mathcal{A}_{\tau_n}$ and $\mathcal{A}_{\tau_{n'}}$ are their $\texttt{LCA}$, and thus we have $\mathcal{A}_{\tau_n} = \mathcal{A}_{\tau_{n'}}$ or one is an ancestor of the other, as proved in the following Lemma.

Lemma 4.1: *Given two nodes n and n', if n and n' are similar, then we have $\mathcal{A}_{\tau_n} = \mathcal{A}_{\tau_{n'}}$, or $\mathcal{A}_{\tau_n}$ is an ancestor or a descendent of $\mathcal{A}_{\tau_{n'}}$.* □

It is very efficient to check whether $\mathcal{A}_{\tau_n} = \mathcal{A}_{\tau_{n'}}$, but it is not easy to check the ancestor-descendant relationship. We propose to use node signatures to address this issue.

Node Signature. Let $\tau_{n,n'} = \min(\tau_n, \tau_{n'})$ denote the smaller depth of nodes $\mathcal{A}_{\tau_n}, \mathcal{A}_{\tau_{n'}}$. If we take the nodes at depth $\tau_{n,n'}$ as a signature of n and n', they must be the same node when n is similar to n'. To generate the signature for all nodes, we set τ_{min} as the minimum depth of their signatures. Then for each node n, we select $\mathcal{A}_{\tau_{min}}$ as its node signature and use this signature to find similar entities.

Example 6: [Node Signature.] Consider three nodes *Computer Science*, *Database*, and *VLDB* in Figure 4. Suppose $\theta = 0.75$. Their τ_n are $\lceil\frac{0.75*2}{2-0.75}\rceil = 2$, $\lceil\frac{0.75*3}{2-0.75}\rceil = 2$, and $\lceil\frac{4*0.75}{2-0.75}\rceil = 3$. Thus their signatures are *Computer Science*, *Computer Science*, and *Database*, respectively. Their node signatures are all *Computer Science*. □

Lemma 4.2: *Given two nodes n and n', if n and n' are similar, their node signatures must be the same node.* □

Example 7: [Signatures for Positive Rules.] Consider e_1 in Figure 1 and the positive rules φ_1^+, φ_2^+ below Example 4. φ_1^+ has only one predicate $f_{ov}(\texttt{Authors}) \geqslant 2$, and the signatures of e_1 *w.r.t.* φ_1^+ are $\texttt{Sig}_{\varphi_1^+}(e_1) = \{$*Xu Chu, John Morcos, Mourad Ouzzani, Paolo Papotti, Ihab F. Ilyas*$\}$. φ_2^+ has two predicates $p_1 : f_{ov}(\texttt{Authors}) \geqslant 1$ and $p_2 : f_{on}(\texttt{Venue}) \geqslant 0.75$. As for p_1, the signatures are all authors. As for p_2, the node signature of *SIGMOD* is *Database*. Thus, the signature set of e_1 *w.r.t.* φ_2^+ is $\texttt{Sig}_{\varphi_2^+}(e_1) = \{$(*Xu Chu, Database*), (*John Morcos, Database*), (*Mourad Ouzzani, Database*), (*Paolo Papotti, Database*), (*Ihab F. Ilyas, Database*), (*Nan Tang, Database*)$\}$. □

Signatures for Negative Rules. We generate the signature set for an entity *w.r.t.* a negative rule similar to the positive rule. The difference is that the thresholds θ_i/σ_i are different. If two entities e and e' have no common signature in every predicate, they must satisfy the negative rule ϕ^-.

C. Filter & Verification for Positive Rules

Filter Strategy for Positive Rules. For each positive rule φ_i^+, we build an inverted index of the signatures of all entities, where each inverted list maintains a mapping from a signature to a list of entities that contain the signature – $L_{\varphi_i^+}(sig)$ contains the set of entities with sig as a signature. Then every entity pair (e, e') on $L_{\varphi_i^+}(sig)$ is a candidate pair. Two entities that are not on the same inverted list cannot be similar – they do not share any common signature.

Verification Acceleration for Positive Rules. It is expensive to verify whether an entity pair satisfies a positive rule (see Section III). To address this issue, we can use transitivity to avoid verifying unnecessary pairs. Given a candidate pair (e, e'), we check whether they are in the same connected component with a constant time complexity[4]. If so, we do not need to verify the pair.

Verification Order of Candidate Pairs. The order of verifying the candidate pairs will affect the performance. If we know that (e, e') match and (e', e'') match, we can infer that (e, e'') match. Naturally, we can infer the answer of (e, e'') based on those of (e, e') and (e', e''). Wang et al. [16] proved that it is optimal to ask the candidate pairs sorted by the probabilities in descending order, and they used the similarity to approximate the probability. The similarity, however, is expensive to compute. Thus, we propose a new method to sort the entity pairs taking both computational cost and similar probability into consideration.

Benefit Order. We compute the *benefit* of verifying a candidate pair. Suppose that each candidate pair (e, e') has a verification cost $\mathcal{C}(e, e')$ and similar probability $\mathcal{P}(e, e')$. We compute the benefit $\mathcal{B}(e, e') = \frac{\mathcal{P}(e,e')}{\mathcal{C}(e,e')}$. Obviously, the greater the benefit of a candidate pair is, the better it is to verify with low cost. Thus we compute the benefit for each pair and verify the candidates sorted by the benefits in a descending order.

Next we discuss how to compute the verification cost and the similar probability.

Verification Cost. The cost of verifying whether (e, e') satisfies a positive rule is the sum of the cost of verifying whether (e, e') can satisfy each predicate. The well-known verification algorithm for edit distance (character-based similarity) is the dynamic-programming algorithm with the time cost of $\mathcal{O}(\theta \min(|e|, |e'|))$, where θ is the similarity threshold. The verification algorithm for Jaccard (set-based similarity) is $\mathcal{O}(|e| + |e'|)$. The verification cost for ontology-based similarity is $O(|d_e| + |d_{e'}|)$ where d_e is the depth of the node in the tree that entity e matches.

Similar Probability. The probability of whether (e, e') can satisfy a positive rule is hard to compute. To address this issue, we can use their shared signatures to approximate the probability, which is the ratio of the number of shared signatures to their average signature number.

[4] We assign each entity a partition ID and utilize a union-find data structure. If e and e' are verified that they satisfy a positive rule, we update their partition ID to the same ID. Assume the partition ID of e is i and that of e' is j, and $i < j$, we change the partition ID of e' to i.

Example 8: [Efficient Checking for Positive Rules.] Given entities in Figure 1 and positive rules φ_1^+, φ_2^+ below Example 4. We generate two candidates $\{(e_1, e_3), (e_2, e_5)\}$ for φ_1^+ and three candidates $\{(e_1, e_2), (e_1, e_3), (e_2, e_3)\}$ for φ_2^+.

Then, we decide the order of verification. The cost of verifying (e_1, e_3) *w.r.t.* φ_1^+ is $\mathcal{C}(e_1, e_3) = 10$, and the similarity probability is $\mathcal{P}(e_1, e_3) = \frac{1}{4} = 0.25$. Thus, the benefit of verifying (e_1, e_3) is 0.025. We compute the benefit for other candidates, and sort them as $\{(e_2, e_5)_{\varphi_1^+}, (e_1, e_3)_{\varphi_1^+}, (e_1, e_3)_{\varphi_2^+}, (e_2, e_3)_{\varphi_2^+}, (e_1, e_2)_{\varphi_2^+}\}$. In this order, we verify $(e_2, e_5), (e_1, e_3)$ for φ_1^+ and (e_2, e_3) for φ_2^+, then e_1, e_2, e_3, e_5 are in the same partition. □

D. Filter & Verification for Negative Rules

Filter Strategy for Negative Rules. After specifying the pivot partition P^*, we utilize the signatures to detect mis-categorized entities from another partition $P \in \mathbf{P}\backslash\{P^*\}$. For each negative rule ϕ_i^-, it generates the signature set $L_{\phi_i^-}(P^*)$ of P^*, and the signature set $L_{\phi_i^-}(P)$ of P. If $L_{\phi_i^-}(P^*) \cap L_{\phi_i^-}(P) = \varnothing$, P is a mis-categorized partition; otherwise, we add $(e \in P, e^* \in P^*)$ into the candidate set.

Verification Acceleration for Negative Rules. It is also inefficient to verify every candidate (e, e^*). Note that once we find a dissimilar pair, P is a mis-categorized partition. In other words, if $\phi^-(e, e^*)$ returns $\mathtt{true}$, there is no need to verify other entities from P. Thus we want to first verify the pair with the smallest probability in order to prune a partition. In addition, the pair (e, e^*) with smaller verification cost $\mathcal{C}(e, e^*)$ should be verified first. So, the benefit should be defined as $\frac{1}{\mathcal{C}(e,e^*)\mathcal{P}(e,e^*)}$. Then we compute the benefit of each entity pair and verify the pairs sorted by the benefit in a descending order.

Example 9: [Efficient Checking for Negative Rules.] Following the above example, partition $P_1 : \{e_1, e_2, e_3, e_5\}$ is regarded as the pivot partition P^*. Consider the negative rules ϕ_1^-, ϕ_2^- below Example 4 and other two partitions $P_2 : \{e_4\}$ and $P_3 : \{e_6\}$. We can conclude that P_2 is mis-categorized partition based on ϕ_1^-, because $L_{\phi_1^-}(P_2) = \{$*Yunqing Xia, NJ Tang, Amir Hussain, Erik Cambria*$\}$ which cannot match any signature of the entities in P^*. In the same way, we can detect that P_3 is also a mis-categorized partition based on ϕ_2^- since $L_{\phi_2^-}(P^*) \cap L_{\phi_2^-}(P_3) = \varnothing$. □

V. Rule Generation

In practice, it is hard for human to fine tune the parameters of all rules, such as similarity functions and their associated thresholds. Hence, we discuss how to generate positive/negative rules from examples. A positive/negative example is a pair of entities that are/aren't in the same category. Note that finding "good examples" is a known challenging problem in training EM models [17], where the good examples normally refer to the non-matched pairs that are likely to be confused as matches. Fortunately, in our case, it is much easier to find good examples, since we only find examples within groups and mis-categorized entities can be paired with any other correctly categorized entities as good examples. Another challenge, which is not well addressed before, is how to derive rules only from examples.

A. Positive Rule Generation

Positive Rule Generation. Given a multi-valued relation $R(A_1, ..., A_m)$, a set of positive examples $\mathbf{S}^+$, a set of negative examples $\mathbf{S}^-$, and a library of similarity functions F, the problem is to find a set Σ^+ of positive rules to maximize a pre-defined objective function $\mathbb{F}(\Sigma^+, \mathbf{S}^+, \mathbf{S}^-)$.

Objective Function. Consider a set of positive rules Σ^+. Given a rule $\varphi^+ \in \Sigma^+$, let $\mathcal{E}_{\varphi^+}$ be the set of entity pairs satisfying φ^+ and $\mathcal{E}_{\Sigma^+} = \bigcup_{\varphi^+ \in \Sigma^+} \mathcal{E}_{\varphi^+}$. To evaluate the quality of Σ^+, we focus on a general case of objective function $\mathbb{F}(\Sigma^+, \mathbf{S}^+, \mathbf{S}^-)$: the larger $|\mathcal{E}_{\Sigma^+} \cap \mathbf{S}^+|$, the larger $\mathbb{F}(\Sigma^+, \mathbf{S}^+, \mathbf{S}^-)$; and the smaller $|\mathcal{E}_{\Sigma^+} \cap \mathbf{S}^-|$, the larger $\mathbb{F}(\Sigma^+, \mathbf{S}^+, \mathbf{S}^-)$. Many functions belong to this general case, *e.g.,* $\mathbb{F}(\Sigma^+, \mathbf{S}^+, \mathbf{S}^-) = |\mathcal{E}_{\Sigma^+} \cap \mathbf{S}^+| - |\mathcal{E}_{\Sigma^+} \cap \mathbf{S}^-|$.

From Infinite to Finite. A naïve method of rule generation is to enumerate all possible rules and then select some of them to maximize the objective function. Then, a problem arises: *Whether the number of all possible positive rules is finite?* The answer is yes. Each rule contains at most m predicates, where m is the number of attributes. For each predicate, there are $|F|$ similarity functions that can be chosen. Although it seems to have infinite similarity thresholds, we can pick finite thresholds which can also maximize the objective function. Next we use an example to illustrate the idea. Consider a rule φ^+ with a single predicate $f(A) \geqslant \theta_1$ and another rule φ'^+ with a single predicate $f(A) \geqslant \theta_2$ by relaxing θ_1 to θ_2 ($\theta_2 < \theta_1$). Obviously, $\mathcal{E}_{\varphi^+} \subseteq \mathcal{E}_{\varphi'^+}$. If there is no positive example in $\mathcal{E}_{\varphi'^+} \backslash \mathcal{E}_{\varphi^+}$, then $\mathbb{F}(\varphi^+, \mathbf{S}^+, \mathbf{S}^-) \geqslant \mathbb{F}(\varphi'^+, \mathbf{S}^+, \mathbf{S}^-)$, and we only need to keep θ_1. Then, a question is: *Which thresholds affect the number of satisfied positive examples?* We compute the similarity for all positive examples on attribute A, and only these similarities can affect the number of satisfied positive examples. Thus, the number of possible thresholds for an attribute is only $|F||\mathbf{S}^+|$.

Theorem 3: *Consider a set of positive examples* $\mathbf{S}^+$*, a set of negative examples* $\mathbf{S}^-$*, an objective function* $\mathbb{F}(\Sigma^+, \mathbf{S}^+, \mathbf{S}^-)$*, and two sets of positive rule* Σ_1^+, Σ_2^+*. Suppose* Σ_2^+ *is transformed from* Σ_1^+ *that only replacing* $f(A) \geqslant \theta_1$ *in* Σ_1^+ *by* $f(A) \geqslant \theta_2$*. If* $\theta_2 < \theta_1$ *and there is no positive example in* $\mathcal{E}_{\Sigma_2^+} \backslash \mathcal{E}_{\Sigma_1^+}$, $\mathbb{F}(\Sigma_2^+, P, N) \leqslant \mathbb{F}(\Sigma_1^+, P, N)$. □

The correctness of theorem has been proved in [3]. Based on this idea, we can generate a finite set of predicates for each attribute. Let $\mathcal{C}_p^+(A)$ denote all candidate predicates of attribute A. For each similarity function $f \in F$ and each pair of positive examples (e, e') from $\mathbf{S}^+$, we compute θ based on $f(A)$ for e and e', and add $f(A) \geqslant \theta$ into $\mathcal{C}_p^+(A)$. For all other thresholds, we can safely prune them since they cannot provide a higher objective value.

Example 10: [Candidate Predicates.] Consider the entities in Figure 1, we have positive/negative examples as follows: $\mathbf{S}^+ = \{(e_1, e_2), (e_1, e_3), (e_1, e_5), (e_2, e_3), (e_2, e_5), (e_3, e_5)\}$; $\mathbf{S}^-=\{(e_1,e_4),(e_1,e_6),(e_2,e_4),(e_2,e_6),(e_3,e_4),(e_3,e_6),(e_4,e_5),(e_5,e_6)\}$. Based on Theorem 3, we have the following predicates. For ease of presentation, we use specific similarity function for each attribute: $\mathcal{C}_p^+(\texttt{Title})= \{f_j(\texttt{Title})\geqslant 0.3, f_j(\texttt{Title})\geqslant 0.07, f_j(\texttt{Title})\geqslant 0.05\}$; $\mathcal{C}_p^+(\texttt{Authors}) = \{f_{ov}(\texttt{Authors}) \geqslant 2, f_{ov}(\texttt{Authors}) \geqslant 1\}$; $\mathcal{C}_p^+(\texttt{Venue}) = \{f_{on}(\texttt{Venue}) \geqslant 0.75, f_{on}(\texttt{Venue}) \geqslant 0.5\}$; □

Based on the finite set of possible predicates on each attribute A_i, we can generate all possible rules by selecting 0-1 predicate from each set $\mathcal{C}_p^+(A_i)$ for $1 \leqslant i \leqslant m$. Let Σ_a^+ denote the set of all possible rules. Then from Σ_a^+, we aim to find a subset to maximize the objective function. We prove that this problem is NP-hard by a reduction from the maximum coverage problem.

Theorem 4: *The rule generation problem is NP-hard.* □

B. An Enumeration Algorithm

Any subset Σ_c^+ of Σ_a^+ is a candidate set of positive rules. We enumerate each subset Σ_c^+, compute its objective-function value, and select the one with the maximal value. Obviously, there are large numbers of candidates, *i.e.,* $2^{|\Sigma_a^+|} = \mathcal{O}(2^{|F|^m |\mathbf{S}^+|^m})$, where m is the number of attributes, $|F|$ is the number of similarity functions, and $|\mathbf{S}^+|$ is the number of positive examples. Clearly, this enumeration-based method is rather expensive.

Example 11: [Enumeration Algorithm.] We enumerate 35 positive rules based on $\mathcal{C}_p^+(\texttt{Title}), \mathcal{C}_p^+(\texttt{Authors})$ and $\mathcal{C}_p^+(\texttt{Venue})$ such as

$\varphi^+ : f_j(\texttt{Title}) \geqslant 0.31 \wedge f_{ov}(\texttt{Authors}) \geqslant 2 \wedge f_{on}(\texttt{Venue}) \geqslant 0.75$
$\varphi^{+\prime} : f_j(\texttt{Title}) \geqslant 0.31 \wedge f_{ov}(\texttt{Authors}) \geqslant 2 \wedge f_{on}(\texttt{Venue}) \geqslant 0.5$
$\varphi^{+\prime\prime} : f_j(\texttt{Title}) \geqslant 0.31 \wedge f_{ov}(\texttt{Authors}) \geqslant 1$ ...

Then, we enumerate all combinations of these positive rules to construct candidate sets of rules. Suppose that Σ_c^+ contains the above three rules and the objective function is $\mathbb{F}(\Sigma^+, \mathbf{S}^+, \mathbf{S}^-) = |\mathcal{E}_{\Sigma^+} \cap \mathbf{S}^+| - |\mathcal{E}_{\Sigma^+} \cap \mathbf{S}^-|$. As $\mathcal{E}_{\Sigma_c^+} = \{(e_1, e_3)\}$, $\mathbb{F}(\Sigma_c^+, \mathbf{S}^+, \mathbf{S}^-) = 1$. For other candidates, we also compute their objective-function value, and then select the one with the maximal value. □

C. A Greedy Algorithm

Due to the high computational complexity of the enumeration algorithm, we propose a greedy algorithm to find a near-optimal set of rules $\hat{\Sigma}^+$. Initially $\hat{\Sigma}^+ = \varnothing$, and we greedily choose the currently best rule until the objective-function value cannot be improved. To generate the best rule φ^+, we greedily select the best predicate as follows.

Given a set of positive examples $\mathbf{S}^+$, a set of negative examples $\mathbf{S}^-$, and sets of candidate predicate $\mathcal{C}_p^+(A_i)$ for each attribute A_i, we first choose the best predicate to initialize the rule $\varphi^+ = f(A) \geqslant \theta$ which has the maximal objective-function value $\mathbb{F}(\varphi^+, \mathbf{S}^+, \mathbf{S}^-)$. Meanwhile, we update the example set $\mathbf{S}^+, \mathbf{S}^-$ to $\mathbf{S}'^+, \mathbf{S}'^-$ by removing the examples that cannot satisfy φ^+.

Afterwards, we pick another predicate p' to join with φ^+ which can prune the negative examples that have satisfied φ^+ as many as possible (smaller $|\mathcal{E}_{p'} \cap \mathbf{S}'^-|$) and keep the positive examples as many as possible (larger $|\mathcal{E}_{p'} \cap \mathbf{S}'^+|$). Thus, we select the one with the maximal objective-function value $\mathbb{F}(p', \mathbf{S}'^+, \mathbf{S}'^-)$ and update the positive rule as $\varphi^+ = f(A) \geqslant \theta \wedge f'(A') \geqslant \theta'$. Another predicate will be chosen until $\mathbb{F}(\varphi^+, \mathbf{S}^+, \mathbf{S}^-)$ cannot be improved and we get φ^+.

We generate a set of rules $\hat{\Sigma}^+$ on the basis of the steps stated above. After φ^+ is added into $\hat{\Sigma}^+$, we should update the example set $\mathbf{S}^+, \mathbf{S}^-$ to $\mathbf{S}''^+, \mathbf{S}''^-$ by removing the examples that satisfy φ^+ and make use of $\mathbf{S}''^+, \mathbf{S}''^-$ to measure the performance of other positive rules. The algorithm terminates when $\mathbb{F}(\Sigma^+, \mathbf{S}^+, \mathbf{S}^-)$ cannot be improved.

Example 12: Consider the entities in Figure 1. The positive examples, negative examples, all predicates and the objective function are the same with the above example.

At first, we compute the score of the objective function for all predicates. Let p_1 denote the predicate $f_{ov}(\texttt{Authors}) \geqslant 2$. As $\mathcal{E}_{p_1} = \{(e_1, e_3), (e_2, e_5)\}$, $\mathbb{F}(p_1, \mathbf{S}^+, \mathbf{S}^-) = 2 - 0 = 2$. Actually, it is the predicate whose objective value is maximum. So we first initialize the rule $\varphi_1^+ = p_1$ and update the example set as $\mathbf{S}'^+ = \{(e_1, e_3), (e_2, e_5)\}$ and $\mathbf{S}'^- = \varnothing$. Since the set of negative examples is empty, we generate one positive rule φ_1^+: $f_{ov}(\texttt{Authors}) \geqslant 2$.

Next, we update the example set by removing the examples that satisfy φ_1^+. Then, $\mathbf{S}''^+ = \{(e_1, e_2), (e_1, e_5), (e_2, e_3), (e_3, e_5)\}$ and $\mathbf{S}''^- = \{(e_1, e_4), (e_1, e_6), (e_2, e_4), (e_2, e_6), (e_3, e_4), (e_3, e_6), (e_4, e_5), (e_5, e_6)\}$. We re-evaluate the values of objective function for all predicates, and generate another positive rule φ_2^+: $f_{ov}(\texttt{Authors}) \geqslant 1 \bigwedge f_{on}(\texttt{Venue}) \geqslant 0.75$. □

D. Negative Rule Generation

The goal of negative rules is to identify mis-categorized partitions. We want to select the negative rules Σ^- to cover more negative examples (larger $|\mathcal{E}_{\Sigma^-} \cap \mathbf{S}^-|$) and less positive examples (smaller $|\mathcal{E}_{\Sigma^-} \cap \mathbf{S}^+|$). The objective function should be $\mathbb{F}(\Sigma^-, \mathbf{S}^+, \mathbf{S}^-) = |\mathcal{E}_{\Sigma^-} \cap \mathbf{S}^-| - |\mathcal{E}_{\Sigma^-} \cap \mathbf{S}^+|$, or other functions that can measure the target. Theorem 3 can be utilized to generate a finite set of predicates for each attribute. Let $\mathcal{C}_p^-(A)$ denote all candidate predicates of attribute A. For each similarity function $f \in F$ and each pair of negative examples (e, e') from $\mathbf{S}^-$, we compute $\sigma = f(e[A], e'[A])$ and add $f(A) \leqslant \sigma$ into $\mathcal{C}_p^-(A)$. Recall that in each step of our greedy algorithm, we generate a negative rule ϕ^- with the maximal score based on $\mathbb{F}(\phi^-, \mathbf{S}^+, \mathbf{S}^-)$. Thus, these rules can be applied in the order of generation.

VI. Experiments

We conduct experiments to answer the following questions for discovering mis-categorized entities: (Exp-1) How good is our approach compared with entity matching solutions? (Exp-2) How good is our approach compared with machine learning approaches? (Exp-3) What is the effect of multiple negative rules? (Exp-4) What is the effect of positive rules? (Exp-5) How efficient is our signature-based algorithm? (Exp-6) How good are the generated rules?

A. Experimental Setup

Datasets. (1) Google Scholar. We crawled 200 Google Scholar pages (*i.e.,* groups), with the average number of 340 entities in a group, all from recent SIGMOD/VLDB/ICDE PC members. Each entity consists of eight attributes: Title, Authors, Date, Venue, Volume, Issue, Pages and Publisher. (2) Amazon Product. We downloaded the data from http://jmcauley.ucsd.edu/data/amazon/links.html [18]. We used 4286 product categories and each product has 8 attributes: Asin, Title, Brand, Also_bought, Also_viewed, Bought_together, Buy_after_viewing and Description.

Mis-categorized Entities. We did not inject mis-categorized entities to Google Scholar because they were dirty originally. Their ground truth was manually verified. For Amazon Product, we injected some products from similar categories to a group as mis-categorized entities for better testing the performance *w.r.t.* different error rates under a more controlled evaluation. The errors were produced with a rate $e\%$, *i.e.,* the percentage of the number of mis-categorized entities over all entities.

Ontologies. Ontologies are not available for all attributes. Fortunately, some are publicly available, *e.g.,* Google Scholar Metrics for Venue (see Section II). For the attributes that do not have ontologies in presence, we can build them. For instance, for product description, we utilized Latent Dirichlet Allocation (LDA) [19] to learn a hierarchy structure as follows. Given a set S of product descriptions, we first learned k topics from S using LDA, and categorized each document to the most likely topic. The set S was divided into $\{S_1, ..., S_k\}$. Then, for each subset S_i, we continued to learn m topics from S_i and divided it into $\{S_{i_1}, ..., S_{i_m}\}$. Then we had three layers: the root was a dummy node; the second layer had k nodes corresponding to the k topics learned from S and the third layer had $k \times m$ nodes learned from each S_i. If there were still many documents in a topic, we would further split it.

Positive/Negative Rules. The positive/negative rules were generated as described in Section V. For Google Scholar, we used two positive rules and three negative rules, learned from 229 positive examples and 201 negative examples.

$\varphi_1^+ : f_{ov}(\texttt{Authors}) \geqslant 2$

$\varphi_2^+ : f_{ov}(\texttt{Authors}) \geqslant 1 \wedge f_{on}(\texttt{Venue}) \geqslant 0.75$

$\phi_1^- : f_{ov}(\texttt{Authors}) = 0$

$\phi_2^- : f_{ov}(\texttt{Authors}) \leqslant 1 \wedge f_{on}(\texttt{Venue}) \leqslant 0.25$

$\phi_3^- : f_{ov}(\texttt{Authors}) \leqslant 1 \wedge f_{on}(\texttt{Title}) \leqslant 0.25$

Three positive rules and two negative rules were applied in Amazon Product, learned using 247 positive examples and 245 negative examples. The entities associated with these training examples were removed for testing data.

$\varphi_3^+ : f_{ov}(\texttt{Also\_bought}) \geqslant 2 \wedge f_{ov}(\texttt{Also\_viewed}) \geqslant 2$

$\varphi_4^+ : f_{ov}(\texttt{Bought\_together}) \geqslant 1 \wedge f_{on}(\texttt{Description}) \geqslant 0.75$

$\varphi_5^+ : f_{ov}(\texttt{Buy\_after\_viewing}) \geqslant 1 \wedge f_{on}(\texttt{Description}) \geqslant 0.75$

$\phi_4^- : f_{ov}(\texttt{Also\_bought}) = 0 \wedge f_{on}(\texttt{Description}) \leqslant 0.5$

$\phi_5^- : f_{ov}(\texttt{Also\_viewed}) = 0 \wedge f_{on}(\texttt{Description}) \leqslant 0.5$

Algorithms. We have implemented the following algorithms. (i) DIME: the basic algorithm in Section III; and (ii) DIME$^+$: the signature-based fast algorithm in Section IV. For comparison, we have implemented (iii) CR [8], which is a collective relational entity linkage solution; and (iv) SVM [20], which is a machine learning (ML) approach for classifying entity. For rule generation, we compared with (v) DecisionTree [21], which is a ML-based rule generation method; and (vi) SIFI [3], a heuristic-based approach that searches optimal similarity functions and thresholds.

Algorithm Parameters. For CR, we picked three thresholds 0.5, 0.6, 0.7 and show the best results. That is, when the minimum distance between two groups was larger than the thresholds, CR stopped merging groups and terminated. We used SVM with linear kernel and balanced class weights to optimize F-measure. DecisionTree was run with maximum depth 4, and for SIFI, we asked an expert to formulate the structure of rules.

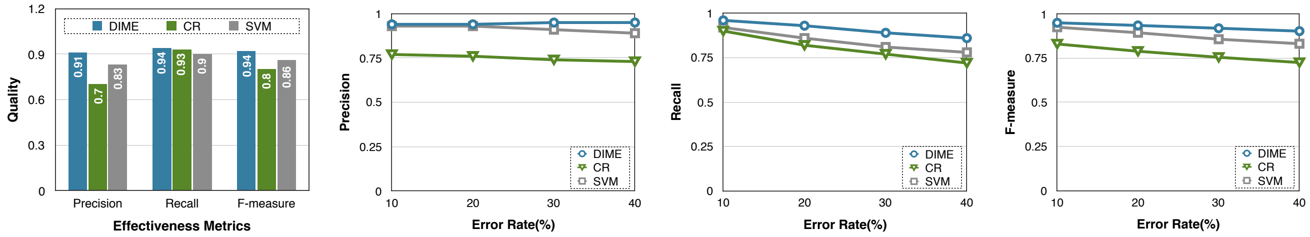
Effectiveness Metrics. We used precision, recall, and F-measure to measure the effectiveness.

Experimental Environment. We implemented DIME, DIME$^+$ and CR in Java, and obtained the Java implementation of SVM from Bilenko et al. [20]. DecisionTree and SIFI were acquired from the authors. All tests were conducted on a PC with a 2.80GHz Intel CPU and 16GB RAM.

B. Experimental Results

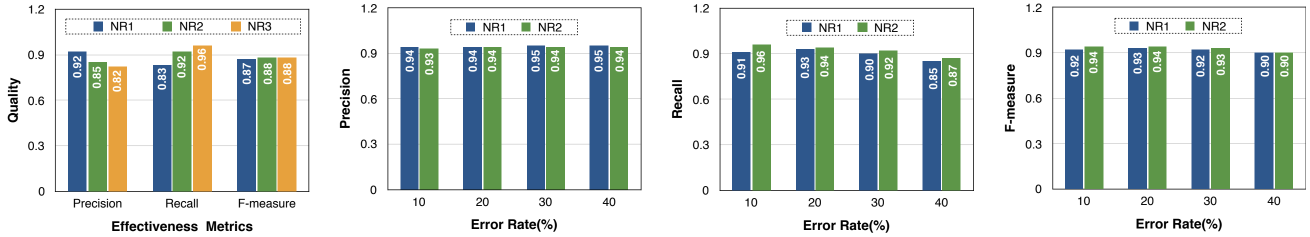
Exp-1: Comparison with EM Approach. We first compared with a collective relational EM solution CR, which was a hierarchical clustering based algorithm that grouped entities based on their distances. In our experiments, we first ran CR to group entities, and then regarded the entities which did not belong to the maximal group as mis-categorized entities. Figure 6 reports the results of comparison about the accuracy. Since Google Scholar was dirty originally, we only varied the error rate of Amazon from 10% to 40%. We used three negative rules for Google Scholar and two negative rules for Amazon. In this part, we report the best precision and recall, when the user dragged the scrollbar. For CR, we tried three thresholds and reported the best.

Figure 6 shows that our method achieved higher precision and recall than CR, and certainly higher F-measure. The reason is that: (1) Some correct entities appeared in small



(a) Scholar (b) Precision (Amazon) (c) Recall (Amazon) (d) F-measure (Amazon)

Figure 6. Compare with State-of-the-art Algorithms



(a) Scholar (b) Precision (Amazon) (c) Recall (Amazon) (d) F-measure (Amazon)

Figure 7. Effectiveness of Scrollbar

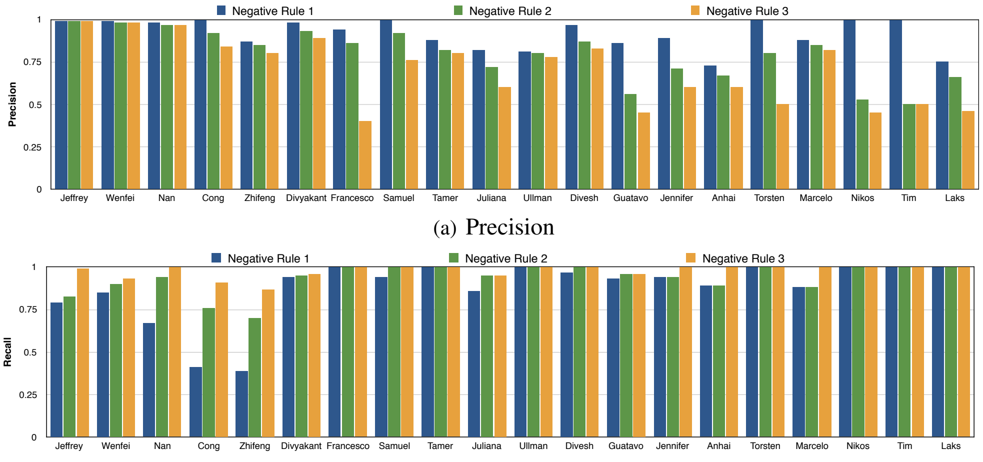
partitions. Without the help of negative rules, these entities were regarded as mis-categorized entities in CR. That is why EM approaches cannot be directly applied to solve our problem. (2) We used ontology similarity to measure the similarity in attribute Venue and Description rather than traditional string similarity, which resulted in higher accuracy. For example, given two entities with venue "International Conference on Very Large Data Bases" and "ACM Transactions on Database Systems" respectively, CR may put them into different groups because of low string similarity. On the other hand, we can judge that these two venues belong to the same field. (3) CR started with the resolutions that only considered attribute string similarity. As an iterative method, one incorrect decision in CR lead to more errors in the following iterations. From the experimental results, we could also find that the clustering based methods for outlier detection cannot be applied to our problem, because they defined the clusters with small sizes to be outliers, but some mis-categorized entities may be in large partitions while some correct entities may appear in small partitions.

Exp-2: Comparison with ML Approach. We also compared with a ML-based method SVM, which has been shown to outperform other machine-learning techniques for classification. We trained two SVM models. The first extracted features from entities and converted each entity to a vector for classification. The positive examples were taken as the entities that should be in the category and the negative examples were mis-categorized entities. In the second one, the features in positive/negative examples were the similarities between two entities that should/shouldn't in the same category. Then it grouped entities and regarded the entities not in the maximal group as mis-categorized entities. Since the similarities between examples were rather important, the latter model was better. Thus we used the latter in SVM. Figure 6 shows that our method had higher precision and recall than SVM in Google Scholar. In Amazon dataset, we achieved a little higher precision than SVM but had much higher recall, which is often favored in such applications of discovering mis-categorized entities. When putting together, our method achieved much higher F-measure. It is noteworthy that when error rate $e\%$ increased, the precision of our method moderately increased. This was because when more mis-categorized entities existed in the group, we found that LDA worked better to differentiate the product description of correct entities from mis-categorized entities. Thus, less false positive would exist. As $e\%$ increased, the recall of all methods decreased. The reason is that, more mis-categorized entities were injected into the dataset as noise which had similar buying behavior and product description. Thus, it became harder to detect them.

Exp-3: Effectiveness of Tuning Negative Rules. In this set of experiments, we studied the effect of tuning negative rules. Recall that we used three negative rules for Google Scholar and two negative rules for Amazon. By default, we showed the user the discovered mis-categorized entities by the first negative rule, and the user can then drag the scrollbar to see the results using other negative rules.

The average results of applying three negative rules for Google Scholar are shown in Figure 7(a), and the results of applying two negative rules for Amazon in different error rates are presented in Figures 7(b)–7(d). The recall value increased when applying more negative rules, which was expected because more mis-categorized entities could be captured. As a trade-off, the precision value decreased, since some correct entities which were not so similar with others were regarded as mis-categorized.

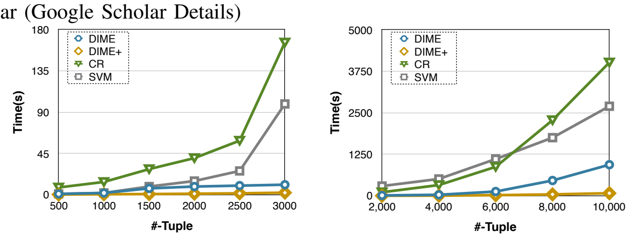
We present the specific results of 20 Google Scholar pages in Figure 8, as different groups have different performances when tuning negative rules. For precision, the first negative rule was the best, which verified that our choice using only author names as the default discriminative attribute in the first negative rule was valid. A further observation is that in most cases, using the default negative rule can get the best precision and close to the best recall. Thus in most cases, the user did not need to touch the scroll bar at all. However, there were several cases, such as Nan Tang, Cong Yu, that needed



(a) Precision

(b) Recall

Figure 8. Effectiveness of Scrollbar (Google Scholar Details)



(a) Time (Scholar)

(b) Time (Amazon)

Figure 9. Efficiency

to use more negative rules to find more mis-categorized entities.It deserves to notice that our tool only suggests mis-categorized entities. It was up to the user to decide which ones to remove from their own group. Hence, the high recall and good precision together made our proposal an ideal tool for cleaning mis-categorized entities.

Exp-4: Effectiveness of Positive Rules. We tested the goodness of our positive rules of grouping entities, which can better understand the reported results in the above experiments. We omitted the result for Amazon as the result was equally good. Table I shows statistics of 20 Google Scholar pages after applying our positive rule to form initial disjoint partitions in step 1 (refer to Figure 5 for more details). The table reads as following, taking Divyakant for instance. Given 652 publications in Divy's Google Scholar page, we computed 104 partitions whose sizes were < 10, the number of total entities in these 104 partitions was 147 that contained 104 mis-categorized entities. Also, there were 2 partitions whose sizes were in $[10, 100)$ that contained 35 entities where 21 were mis-categorized entities. Moreover, there were 2 partitions whose sizes were in $[100, 1000)$ that contained 480 entities without any mis-categorized entities. All numbers about mis-categorized entities were highlighted in red. Table I first tells us that most mis-categorized entities appeared in small partitions, which verified the effectiveness of our conservative positive rule that successfully isolated them. This also helps to explain why our negative rules can discover mis-categorized entities in the above experiments.

Exp-5: Efficiency Study. We compared the efficiency of our algorithms with two baselines `CR` and `SVM`. The tested Google Scholar page contained at most 3000 entities, we sampled six `Google Scholar` pages by varying the number of tuples from 500 to 3000. For `Amazon`, we picked five categories with error rate 40% and varied the number of entities from 2000 to 10000. Figure 9 shows the running time. Our algorithms were faster than `CR` and `SVM`, because in each iteration, `CR` re-evaluated the attribute distance and reference distance between two groups and selected the closest pair to merge. It was time-consuming, especially the size of dataset was large. As for `SVM`, we used part of the dataset as training data and the remaining dataset as testing data. It took lots of iterations for many entities.

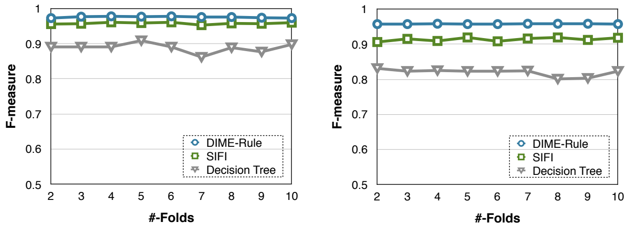
Impact of Indices. With the signature-based framework, the efficiency was significantly improved. As shown in Figure 9, $\texttt{DIME}^+$ was 2-10x faster than `DIME`. For `Google Scholar`, when there were 3000 entities, the runtime of `DIME` was 11 seconds, but 2.3 seconds for $\texttt{DIME}^+$. For `Amazon`, when there were 10000 entities, the runtime of `DIME` was 936 seconds, but 77 seconds for $\texttt{DIME}^+$. The results showed the superiority of our pruning techniques. In reality, the size of group is unlikely to be so large after a former cluster algorithm. But we still utilized a data generator `DBGen` (http://www.cs.utexas.edu/users/ml/riddle/data) to generate some large groups with the number of entities from 20k to 100k to test the efficiency of our algorithms. We used two positive rules and two negative rules and the results are shown in the table below. Our signature-based algorithm $\texttt{DIME}^+$ can run for 100k entities in 175s, which was 15x faster than `DIME`.

	`Gen(20k)`	`Gen(40k)`	`Gen(60k)`	`Gen(80k)`	`Gen(100k)`
`DIME`	430	619	891	1723	2610
$\texttt{DIME}^+$	39	52	69	133	175

Exp-6: Comparison with Rule Generation Methods. We compared our greedy rule generation algorithm `DIME-Rule` with existing ML methods `DecisionTree` and heuristic-based approach `SIFI`. We made cross validation on the training set, and Figure 10 reports F-measure values by varying the number of folds. Note that for `SIFI`, we asked an expert to formulate the structure of rules who was familiar with similarity functions and our datasets. Figure 10 shows that our method can get higher F-measure than `DecisionTree` and `SIFI`. `DecisionTree` failed to find the optimal similarity functions with considerable depth when

Partition Size	[1, 10) #-groups	 #-entities	 #-errors	[10, 100) #-groups	 #-entities	 #-errors	[100, 1000) #-groups	 #-entities	 #-errors
Divyakant	104	147	**104**	2	35	**21**	2	480	0
Jeffrey	1608	2334	**2234**	8	158	**148**	2	508	**199**
Wenfei	224	348	**307**	3	138	0	0	0	0
Nan	51	65	**55**	4	52	**11**	0	0	0
Cong	50	78	**46**	2	67	0	0	0	0
Zhifeng	30	48	**23**	1	50	0	0	0	0
Francesco	60	74	**29**	2	45	0	1	111	0
Samuel	64	93	**18**	3	136	0	0	0	0
Tamer	123	173	**28**	4	75	0	1	109	0
Juliana	64	93	**21**	3	47	0	1	137	0
Ullman	177	220	**40**	5	97	0	1	223	0
Anhai	36	45	**9**	2	102	0	0	0	0
Divesh	67	95	**29**	2	50	0	1	369	0
Gustavo	116	161	**27**	8	268	0	0	0	0
Jennifer	71	88	**18**	2	73	0	1	162	0
Torsten	20	30	**4**	1	50	0	0	0	0
Marcelo	62	107	**8**	3	95	0	0	0	0
Nikos	41	76	**9**	1	55	0	0	0	0
Tim	10	10	**3**	1	75	0	0	0	0
Laks	64	99	**6**	4	58	0	1	96	0

Table I
EFFECT OF POSITIVE RULES



(a) F-measure (Scholar)   (b) F-measure (Amazon)

Figure 10. Effectiveness of Rule Generation

there were a lot of options, and in SIFI, it was hard for expert to always provide the optimal structure of rules and suitable similarity functions.

VII. CONCLUDING REMARKS

We have proposed a new problem that discovers miscategorized entities. We have presented a general rule-based framework to solve this problems in different applications. We have also proposed a signature-based algorithm to efficiently apply the rules. We have demonstrated both the effectiveness and efficiency of our approach on real datasets.

ACKNOWLEDGMENT

The work was supported by the 973 Program of China (2015CB358700), NSF of China (61632016, 61472198, 61521002), and TAL education.

REFERENCES

[1] S. Hao, Y. Xu, N. Tang, G. Li, and J. Feng, "Cleaning your wrong google scholar entries," in *ICDE demo*, 2018.

[2] A. K. Elmagarmid, P. G. Ipeirotis, and V. S. Verykios, "Duplicate record detection: A survey," *IEEE Trans. Knowl. Data Eng.*, vol. 19, no. 1, pp. 1–16, 2007.

[3] J. Wang, G. Li, J. X. Yu, and J. Feng, "Entity matching: How similar is similar," *PVLDB*, 2011.

[4] R. Singh, V. V. Meduri, A. K. Elmagarmid, S. Madden, P. Papotti, J. Quiané-Ruiz, A. Solar-Lezama, and N. Tang, "Synthesizing entity matching rules by examples," *PVLDB*, 2017.

[5] I. Fellegi and A. Sunter, "A theory for record linkage," *Journal of the American Statistical Association*, vol. 64 (328), 1969.

[6] P. Singla and P. Domingos, "Entity resolution with markov logic," in *ICDM*, 2006.

[7] J. Wang, T. Kraska, M. J. Franklin, and J. Feng, "Crowder: Crowdsourcing entity resolution," *PVLDB*, 2012.

[8] I. Bhattacharya and L. Getoor, "Collective entity resolution in relational data," *TKDD*, vol. 1, no. 1, p. 5, 2007.

[9] Y. Li, C. Wang, F. Han, J. Han, D. Roth, and X. Yan, "Mining evidences for named entity disambiguation," in *SIGKDD*, 2013.

[10] R. C. Bunescu and M. Pasca, "Using encyclopedic knowledge for named entity disambiguation," in *EACL*, 2006.

[11] D. Ramage, D. L. W. Hall, R. Nallapati, and C. D. Manning, "Labeled LDA: A supervised topic model for credit attribution in multi-labeled corpora," in *EMNLP*, 2009.

[12] A. K. Jain, "Data clustering: 50 years beyond k-means," *Pattern Recognition Letters*, vol. 31, no. 8, pp. 651–666, 2010.

[13] E. Forgey, "Cluster analysis of multivariate data: Efficiency vs. interpretability of classification," *Biometrics*, vol. 21, p. 768, 1965.

[14] M. Hazman, S. R. El-Beltagy, and A. Rafea, "A survey of ontology learning approaches," *database*, vol. 7, p. 6, 2011.

[15] Y. Jiang, G. Li, J. Feng, and W. Li, "String similarity joins: An experimental evaluation," *PVLDB*, 2014.

[16] J. Wang, G. Li, T. Kraska, M. J. Franklin, and J. Feng, "Leveraging transitive relations for crowdsourced joins," in *SIGMOD*, 2013.

[17] S. Sarawagi and A. Bhamidipaty, "Interactive deduplication using active learning," in *KDD*, 2002.

[18] J. McAuley, C. Targett, Q. Shi, and A. Van Den Hengel, "Image-based recommendations on styles and substitutes," in *SIGIR*, 2015.

[19] R. Das, M. Zaheer, and C. Dyer, "Gaussian LDA for topic models with word embeddings," in *ACL*, 2015.

[20] M. Bilenko and R. J. Mooney, "Adaptive duplicate detection using learnable string similarity measures," in *SIGKDD*, 2003.

[21] C. Gokhale, S. Das, A. Doan, J. F. Naughton, N. Rampalli, J. Shavlik, and X. Zhu, "Corleone: Hands-off crowdsourcing for entity matching," in *SIGMOD*, 2014.